9 mcmctree

Overview

The program `mcmctree` may be the first Bayesian phylogenetic program (Yang and Rannala 1997; Rannala and Yang 1996) , but was very slow and decommissioned since the release of MrBayes (Huelsenbeck and Ronquist 2001).

Since PAML version 3.15 (2005), mcmctree implements the MCMC algorithms of Yang and Rannala (2006) and then of Rannala and Yang (2007) for estimating species divergence times on a given rooted tree using multiple fossil calibrations. This is similar to the multidivtime program of Jeff Thorne and Hiro Kishino. The differences between the two programs are discussed by Yang and Rannala (2006) and Yang (2006, Section 7.4); see also below.

Please refer to any book on Bayesian computation, for example, Chapter 5 in Yang (2006) for the basics of MCMC algorithms.

Here are some notes about the program.

- Before starting the program, resize the window to have 100 columns instead of 80. (On Windows XP/Vista, right-click the command prompt window title bar and change Properties - Layout - Window Size - Width.)

- The tree, supplied in the tree file, must be a rooted binary tree: every internal node should have exactly two daughter nodes. You should not use a consensus tree with polytomies for divergence time estimation using MCMCTREE. Instead you should use a bifurcating ML tree or NJ tree or traditional morphology tree. Note that a binary tree has a chance of being correct, while a polytomy tree has none.

- The tree must not have branch lengths. For example, ((a:0.1, b:0.2):0.12, c:0.3) '>0.8<1.0'; does not work, while ((a, b), c) '>0.8<1.0'; is fine.

- Under the relaxed-clock models (clock = 2 or 3) and if there is no calibration on the root, a loose upper bound (maximal age constraint) must be specified in the control file (RootAge). (There should be no need to use RootAge if clock = 1, but the program insists that you have it. I will try to fix this.)

- *Choice of time unit.* The time unit should be chosen such that the node ages are about 0.01-10. If the divergence times are around 100-1000MY, then 100MY may be one time unit. The priors on times and on rates and the fossil calibrations should all be specified based on your choice of the time scale. For example, if one time unit is 10MY, the following

```
  rgene_gamma = 2 20        * gamma prior for overall rates for genes
sigma2_gamma = 10 100       * gamma prior for sigma^2  (for clock=2 or 3)
```

  means an overall average rate of 2/20 = 0.1 substitutions per site per 10MY or 10^{-8} substitutions per site per year, which is reasonable for mammalian mitochondrial genes. The gamma prior here has the shape parameter $\alpha = 2$, and is fairly diffuse. If you change the time unit, you should keep the shape parameter fixed and change the scale parameter β to have the correct mean. In other words, to use one time unit to represent 100MY, the above should become

```
  rgene_gamma = 2 2         * gamma prior for overall rates for genes
sigma2_gamma = 10 100       * gamma prior for sigma^2  (for clock=2 or 3)
```

Note that the change of the time unit should not lead to a change of the prior for σ^2, because σ^2 is the variance of the log rate: the variance of the logarithm of the rate does not change when you rescale the rate by a constant. When you change the time unit, the fossil calibrations in the tree file should be changed accordingly. While ideally one would want the biological results to be unchanged when one changes the time unit, we know that two components of the model are not invariant to the time scale: the log normal distribution for rates and the birth-death model for times. Nevertheless, tests by Mathieu Groussin did suggest that the choice of time scale has very minimal effects on the posterior time and rate estimates.

- *Specifying the prior on rates.* Choose α for `rgene_gamma` (μ) based on how confident you are about the overall rate. For example, $\alpha = 1$, 1.5, or 2 mean quite diffuse (uninformative) priors. Then adjust β so that the mean α/β is reasonable. To get a rough mean for the overall rate, you can use a few point calibrations to run the ML program baseml or codeml under a strict clock (clock = 1). For example, if a node has the calibration B(0.06, 0.08), you can fix the node age at 0.07 when you run baseml/codeml. If you are analyzing multiple loci/partitions, which have quite different rates, you can use an intermediate value, or the mean or median among the locus rates. The program uses the same prior for μ for all loci.

- It is important that you run the same analysis at least twice to confirm that the different runs produced very similar (although not identical) results. It is critical that you ensure that the acceptance proportions are neither too high nor too low. See below about the variable finetune.

- It is important that you run the program without sequence data (usedata = 0) first to examine the means and CIs of divergence times specified in the prior. In theory, the joint prior distribution of all times should be specified by the user. Nevertheless it is nearly impossible to specify such a complex high-dimensional distribution. Instead the program generates the joint prior by using the calibration distributions and the constaint on the root as well as the birth-death process model to generate the joint prior. This the prior used by the program in the dating analysis. You have to examine it to make sure it is sensible, judged by your knowledge of the species and the relevant fossil record. If necessary, you may have to change the fossil calibrations so that the prior look reasonable.

- The program right now does a simple summary of the MCMC samples, calculating the mean, median and the 95% CIs for the parameters. If you want more sophisticated summaries such as 1-D and 2-D density estimates, you can run a small program ds at the end of the mcmctree run, by typing ds mcmc.out.

- To use hard bounds, you can specify the tail probabilities as 10^{-300} instead of the default 0.025. See table 8 below.

Fossil calibration

Fossil calibration information, in the form of statistical distributions of divergence times (or ages of nodes in the species tree), is specified in the tree file. See table 8 for a summary. Here "fossil" means any kind of external calibration data, including geological events. For a sensible analysis, one should have at least one lower bound and at least one upper bound on the tree, even though they may not be on the same node. The gamma, skew normal, and skew *t* distributions can act as both bounds, so one such calibration is enough to anchor the tree to enable a sensible analysis.

Table 8. Calibration distributions

Calibration	#*p*	Specification	Density
L (lower or minimum bound)	4	'>0.06' or 'L(0.06)' or 'L(0.06, 0.2)' or 'L(0.06, 0.1, 0.5)' or 'L(0.06, 0.1, 0.5, 0.025)'	L(t_L, p, c, p_L) specifies the minimum-age bound t_L, with offset p, and scale parameter c, and left tail probability p_L. The default values are p = 0.1, c = 1, and p_L = 0.025, so >0.06 or L(0.06) means L(0.06, 0.1, 1, 0.025), and L(0.06, 0.2) means L(0.06, 0.2, 1, 0.025). If you would like the minimum bound to be hard, use p_L = 1e-300, but do not use p_L = 0. In other words, use L(0.06, 0.2, 1, 1e-300), not L(0.06, 0.2, 1, 0).
U (upper or maximum bound)	2	'<0.08' or 'U(0.08)' or 'U(0.08, 0.025)'	Eq. 16 & fig. 2b in YR06. U(t_U, p_R) specifies the maximum-age bound t_U, with right tail probability p_R. The default value is p_U = 0.025, so <0.08 or U(0.08) means U(0.08, 0.025). For example U(0.08, 0.1) means that there is 10% probability that the maximum bound 8MY is violated (i.e., the true age is older than 8MY).
B (lower & upper bounds or minimum & maximum bounds)	4	'>0.06<0.08' or 'B(0.06, 0.08)' or 'B(0.06, 0.08, 0.025, 0.025)'	Eq. 17 & fig. 2c in YR06 B(t_L, t_U, p_L, p_U) specifies a pair bound, so that the true age is between t_L and t_U, with the left and right tail probabilities to be p_L and p_U, respectively. The default values are p_L = p_U = 0.025.
G (Gamma)	2	'G(alpha, beta)'	Eq. 18 & fig. 2d in YR06
SN (skew normal)	3	'SN(location, scale, shape)'	Eq. 2 & plots below
ST (skew *t*)	4	'ST(location, scale, shape, df)'	Eq. 4 & plots below
S2N (skew 2 normals)	7	'SN2(p_1, loc1, scale1, shape1, loc2, scale2, shape2)'	p_1: 1 – p_1 mixture of two skew normals.

Note .— #*p* is the number of parameters in the distribution, to be supplied by the user. Figure 2 in YR06 (Yang and Rannala 2006) is figure 7.11 in Yang (2006).

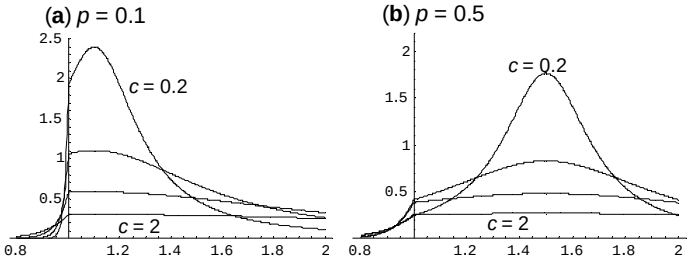
(1) Lower bound (minimal age) is specified as '>0.06' or 'L(0.06)', meaning that the node age is at least 6MY. Here we assume that one time unit is 100 million years. In PAML version 4.2, the implementation of the minimum bound has changed. Instead of the improper soft flat density of Figure 2a in Yang and Rannala (2006) or figure 7.11a in Yang (2006), a heavy-tailed density based on a truncated Cauchy distribution is now used (Inoue et al. 2010). The Cauchy distribution with location parameter $t_L(1 + p)$ and scale parameter ct_L is truncated at t_L, and then made soft by adding α_L = 2.5% of density mass left of t_L. The resulting distribution has mode at $t_L(1 + p)$. The α_L = 2.5% limit is of course at t_L and the 97.5% limit is at

$$t_{97.5\%} = t_L[1 + p + c\cot(\tfrac{\pi A \alpha_R}{1-\alpha_L})],$$

where α_R = 1 – 0.975 and $A = \frac{1}{2} + \frac{1}{\pi}\tan^{-1}\left(\frac{p}{c}\right)$. This is slightly more general than the formula in the paragraph below equation (26) in (Inoue et al. 2010), in that α_L and α_R are arbitrary: to get the 99%

limit when t_L is a hard minimum bound, use $\alpha_L = 0$ and $\alpha_R = 0.01$ so that $t_{99\%} = t_L[1 + p + c \cot(0.01\pi A)]$.

If the minimum bound t_L is based on good fossil data, the true time of divergence may be close to the minimum bound, so that a small p and small c should be used. It is noted that c has a greater impact thatn p on posterior time estimation. The program uses the default values $p = 0.1$ and $c = 1$. However, you are advised to use different values of p and c for each minimum bound, based on a careful assessment of the fossil data on which the bound is based. Below are a few plots of this density. The minimum bound is fixed at $t_L = 1$, but one time unit can mean anything like 100Myr or 1000Myr. For each value of p (0.1 and 0.5), the four curves correspond to $c = 0.2, 0.5, 1, 2$ (from top to bottom nearthe peak). The 2.5% limit is at 1, while the 97.5% limits for those values of c are 4.93, 12.12, 24.43, 49.20, respectively, when $p = 0.1$, and are 4.32, 9.77, 20.65, 44.43 when $p = 0.5$. (Note that those values were incorrectly calculated in Inoue et al. 2010)



(2) Upper bound (maximal age) is specified as '<0.08' or 'U(0.08)', meaning that the node age is at most 8MY.

(3) Both lower and upper bounds on the same node are specified as '>0.06<0.08' or 'B(0.06, 0.08)', meaning that the node age is between 6MY and 8MY.
Note that in all the above three calibrations (L, U, B), the bounds are soft, in that there is a 2.5% probability that the age is beyond the bound (see figure 2 in Yang and Rannala 2006; or figure 7.11 in Yang 2006).

(4) The gamma distribution. 'G(188, 2690)' specifies the gamma distribution with shape parameter $\alpha = 188$ and scale parameter $\beta = 2690$. This has the mean $\alpha/\beta = 0.07$ and the 2.5 and 97.5 percentiles at ### and ###. In earlier versions (3.15, 4a & 4b), the gamma was specified as '`>.06=0.0693<.08`', but this format is not used anymore.

```
((((human, (chimpanzee, bonobo)) 'G(188, 2690)', gorilla), (orangutan, sumatran))
'>.12<.16', gibbon);
```

In the tree above, the human-chimp divergence time has a gamma distribution G(188, 2690), while the orang-utan divergence time has soft bounds between 12MY and 16MY.

The above tree can be read in TreeView, with the calibration information in quotation marks treated as node labels.

You can use the MS Excel function GAMMADIST(X, alpha, beta, 0) to calculate and plot the density function (pdf) of the gamma distribution, and the function GAMMAINV(0.025, alpha, beta) to calculate the 2.5% percentile. However, note that beta in Excel is $1/\beta$ in MCMCTREE (and other PAML programs). In other words, the mean is α/β in MCMCTREE and $\alpha\beta$ in Excel.

(5) Skew normal distribution SN(location, scale, shape) or SN(ξ, ω, α) (Azzalini and Genton 2007). The basic form of the skew normal distribution has density

$$f(z;\, \alpha) = 2\phi(z)\Phi(\alpha z), \tag{1}$$

where $\phi()$ and $\Phi()$ are the PDF and CDF of the standard normal distribution respectively. Then $x = \xi + \omega z$, has the skew normal distribution SN(ξ, ω, α) with location parameter ξ, scale parameter ω, and shape parameter α. The density is

$$f_{\mathrm{SN}}(x;\, \xi,\, \omega,\, \alpha) = \frac{1}{\omega} \times \frac{1}{\sqrt{2\pi}} \mathrm{e}^{-\frac{(x-\xi)^2}{2\omega^2}} \int_{-\infty}^{\alpha\left(\frac{x-\xi}{\omega}\right)} \frac{1}{\sqrt{2\pi}} \mathrm{e}^{-\frac{u^2}{2}} \mathrm{d}u\,. \tag{2}$$

for $-\infty < \xi < \infty$, $0 < \omega < \infty$, $-\infty < \alpha < \infty$. Let $\delta = \alpha \big/ \sqrt{1+\alpha^2}$. The mean and variance are

$$\begin{aligned} E(x) &= \xi + \omega\delta\sqrt{2/\pi}\,, \\ \mathrm{Var}(x) &= \omega^2\left(1 - \tfrac{2\delta^2}{\pi}\right). \end{aligned} \tag{3}$$

(6) **Skew *t* distribution, ST(location, scale, shape, df) or ST(ξ, ω, α, ν)** (Azzalini and Genton 2007), with location parameter ξ, scale parameter ω, shape parameter α, and degree of freedom ν, has density

$$f_{\mathrm{ST}}(x;\xi,\omega,\alpha,\nu) = \frac{2}{\omega} t(z;\nu) T\left(\alpha z\sqrt{(\nu+1)\big/(\nu+z^2)};\nu+1\right), \tag{4}$$

where $z = (x - \xi)/\omega$, t and T are the PDF and CDF of the standard t distribution, respectively. These are defined as follows.

$$\begin{aligned} t(z;\nu) &= \frac{\Gamma\left(\frac{1}{2}(\nu+1)\right)}{\sqrt{\pi\nu}\,\Gamma\left(\frac{1}{2}\nu\right)}\left[1+\frac{z^2}{\nu}\right]^{-(\nu+1)/2}, \\ T(z;\nu) &= \begin{cases} \frac{1}{2} I_{\nu/(\nu+z^2)}\left(\frac{1}{2}\nu, \frac{1}{2}\right), & \text{if } z < 0, \\ 1 - T(-z;\nu), & \text{if } z \ge 0. \end{cases} \end{aligned} \tag{5}$$

where $\Gamma()$ is the gamma function, and

$$I_p(a,b) = \frac{1}{B(a,b)} \int_0^p u^{a-1}(1-u)^{b-1}\,\mathrm{d}u \tag{6}$$

is the incomplete beta function ratio, or the CDF of the beta(a, b) distribution, while

$$B(a,b) = \int_0^1 u^{a-1}(1-u)^{b-1}\,\mathrm{d}u = \frac{\Gamma(a)\Gamma(b)}{\Gamma(a+b)} \tag{7}$$

is the beta function.

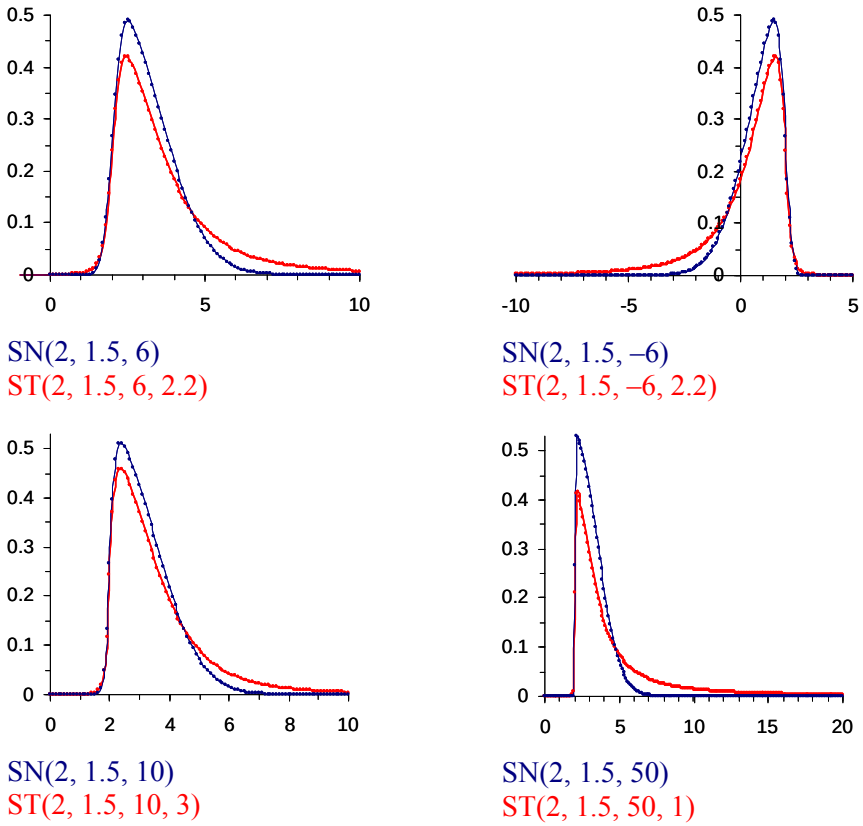


Figure 1. Density functions for skew normal (blue) and skew *t* (red) distributions. Skew *t* has heavier tails than skew normal.

Here are a few notes about the skew normal and skew *t* distributions.

- When the shape parameter $\alpha = 0$, the distributions become the standard (symmetrical) normal and *t* distributions.
- Changing α to $-\alpha$ flips the density at $x = \xi$ (the location parameter). Fossil calibrations should have long right tails, which means $\alpha > 0$.
- A larger $|\alpha|$ means more skewed distributions. When $|\alpha| = \infty$, the distribution is called folded normal or folded *t* distribution, that is, the normal or *t* distribution truncated at ξ from the left ($\alpha = \infty$) or from the right ($\alpha = -\infty$).
- When the degree of freedom $\nu = \infty$, the *t* or skew *t* distribution becomes the normal or skew normal distribution. The smaller ν is, the heavier the tails are. A small ν (1-4 , say) with a large shape parameter α in the skew *t* distribution represents virtually hard minimal bound and very uncertain maximal bound. When $\nu = 1$, the *t* distribution is known as Cauchy distribution, which does not have mean or variance.
- Both skew normal and skew *t* distributions go from $-\infty$ to ∞. In MCMCTREE, negative

values are automatically truncated, so only the positive part is used. If feasible, try to construct the distribution so that the probability for negative values is small (<0.1%, say).

- Please visit the web site http://azzalini.stat.unipd.it/SN/ to plot skew normal and skew *t* distributions. R routines are also available for such plots. The equations above are for my testing and debugging. I think I should remove them later on.

The control file

You can use the files in the folder examples/SoftBound/ to duplicate the results of Yang and Rannala (2006: table 3) and Rannala and Yang (2007: table 2). Below is a copy of the control file mcmctree.ctl.

```
          seed = -1234567
       seqfile = mtCDNApri123.txt
      treefile = mtCDNApri.trees
       outfile = out

         ndata = 3
       usedata = 1    * 0: no data; 1:seq like; 2:use in.BV; 3: out.BV
         clock = 1    * 1: global clock; 2: independent rates; 3: correlated rates
       RootAge = '>0.8<1.2'

         model = 0    * 0:JC69, 1:K80, 2:F81, 3:F84, 4:HKY85
         alpha = 0    * alpha for gamma rates at sites
         ncatG = 5    * No. categories in discrete gamma

     cleandata = 0    * remove sites with ambiguity data (1:yes, 0:no)?
 BlengthMethod = 0    * 0: arithmetic; 1: geometric; 2: Brownian

       BDparas = 1 1 0    * birth, death, sampling
   kappa_gamma = 6 2      * gamma prior for kappa
   alpha_gamma = 1 1      * gamma prior for alpha

   rgene_gamma = 2 2     * gamma prior for rate for genes
  sigma2_gamma = 1 1    * gamma prior for sigma^2     (for clock=2 or 3)

      finetune = 0.04 0.02 0.3 0.1 0.3 * time, rate, mixing, paras, RateParas

         print = 1
        burnin = 10000
      sampfreq = 10
       nsample = 10000
```

seed should be assigned a negative or positive integer. A negative integer (such as –1) means that the random number seed is determined from the current clock time. Different runs will start from different places and generate different results due to the stochastic nature of the MCMC algorithm. You should use this option and run the program at least twice, to confirm that the results are very similar between runs (identical to 1MY or 0.1MY, depending on the desired precision). If you obtain intolerably different results from different runs, you obviously won't have any confidence in the results. This lack of consistency between runs can be due to many reasons: including slow convergence, poor mixing, insufficient samples taken, or errors in the program. Thus you can check to make sure (i) that the chain is at reasonably good place when it reached 0% (the end of burn-in), indicating that the chain may have converged; (ii) that the acceptance proportion of all proposals used by the algorithm are neither too high nor too low (see below about **finetune**) indicating that the chain is mixing well; (iii) that you have taken enough samples (see **nsample** and **burnin** below). If you give **seed** a positive number, that number will be used as the real seed. Then running the program multiple times will produce exactly the same results. This is useful for debugging the program and should not be the default option for real data analysis.

ndata is the number of loci (or site partitions) in a combined analysis. The program allows some species to be missing at some loci. The mt primate data included protein-coding genes, and the three codon positions are treated as three different partitions. In the combined analysis of multiple

gene loci, the same substitution model is used, but different parameters are assigned and estimated for each partition.

usedata. 0 means that the sequence data will not be used in the MCMC, with the likelihood set to 1, so that the MCMC approximates the prior distribution. This option is useful for testing and debugging the program, and is also useful for generating the prior distribution of the divergence times. The fossil calibrations and the constraints on the root you specify are not the real prior that is implemented in the program; for example, they may not even satisfy the requirement that ancestors should be older than descendents. The prior that is used by the program can be generated by running the chain without data. usedata = 1 means that the sequence data will be used in the MCMC, with the likelihood calculated using the pruning algorithm of Felsenstein (1981), which is exact but very slow except for very small species trees. This option is available for nucleotide sequences only, and the most complex model available is HKY85+Γ. usedata = 2 and 3 implement a method of approximate likelihood calculation. They can be used to analyze nucleotide, amino acid, and codon sequences, using nucleotide, amino acid, and codon substitution models, respectively.

usedata = 2 inBVfilename specifies the approximate likelihood calculation, with the input (gradient & Hessian matrix etc.) in the file.

clock. The clock variable is used to implement three models concerning the molecular clock: 1 means global molecular clock, so that the rate is constant across all lineages on the tree (even though the rate may vary among multiple genes); 2 means the independent-rates model, and 3 the auto-correlated rates model. See Rannala and Yang (2007) and Section §7.4 in Yang (2006) for details.

RootAge. The RootAge variable is used to specify a loose constaint on the age of the root, to constrain the root age from above. It is used if no such constraint is available through a fossil calibration on the root. Note that fossil calibrations are specified in the tree file. Two formats are accepted, specifying either a maximum bound (e.g., RootAge = '<1.2') or a pair of minimum and maximum bounds (e.g., RootAge = '>0.8<1.2'). The RootAge variable is ignored if a fossil calibration on the root is specified in the tree file in the form of a maximum bound, a pair of minimum and maximum bounds, or a gamma distribution. If the fossil calibration in the tree file is a minimum bound on the root (e.g. '>0.9'), and you specify RootAge = '<1.2', then the program implements the pair of bounds, equivalent to specifying the calibration '>0.9<1.2' on the root.

model, alpha, ncatG are used to specify the nucleotide substitution model. These are the same variables as used in baseml.ctl. If alpha ≠ 0, the program will assume a gamma-rates model, while alpha = 0 means that the model of one rate for all sites will be used. Those variables have no effect when usedata = 2.

cleandata = 0 means that alignment gaps and ambiguity characters will be treated as missing data in the likelihood calculation (see pages 107-108 in Yang 2006). = 1 means that any sites at which at least one sequence has an alignment gap or ambiguity character will be deleted before analysis. This variable is used for usedata = 1 and 3 and has no effect if usedata = 2.

BDparas = 2 2 .1 specifies the three parameters (birth rate λ, death rate μ and sampling fraction ρ) in the birth-death process with species sampling (Yang and Rannala 1997), which is used to specify the prior of divergence times (Yang and Rannala 2006). The node times are order statistics from a kernel density, which is specified by those parameters. A few kernel densities are shown in figure 2 of Yang and Rannala (1997) or figure 7.12 in Yang (2006). The Mathematica code for plotting the density for given parameters λ, μ and ρ is posted at the web site http://abacus.gene.ucl.ac.uk/ziheng/data.html. By adjusting parameters λ, μ and ρ to generate different tree shapes, one can assess the impact of the prior on posterior divergence time estimation.

Intuitively, the node ages and thus the shape of the tree are determined by the parameters as follows. There are $s - 1$ internal nodes and thus $s - 1$ node ages in the rooted tree of s species. The age of the root is fixed, so the $s - 2$ node ages are relative to the root age (they are all between 0 and 1). We draw $s - 2$ independent random variables from the kernel density and order them. Those ordered variables will then be the node ages. Thus if the kernel density has the L shape, all internal nodes tend to be young (relative to the root), and the tree will have long internal branches and short tip branches. In contrast, if the kernel density has the reverse L shape, the node ages are large and the nodes close to the root, then the tree will be bush-like. See pages 250-251 in Yang (2006). (Strictly speaking the above description is accurate if fossil calibration is available for the root only but not for any other nodes. Otherwise the kernel density specifies the distribution of the ages of non-calibration nodes only, and the impact of the kernel on the joint distribution of all node ages may be complex, depending on the locations of the calibration nodes.)

kappa_gamma = 6 2 specifies the shape and scale parameters (α and β) in the gamma prior for parameter κ (the transition/transversion rate ratio) in models such as K80 and HKY85. This has no effect in models such as JC69, which does not have the parameter. Note that the gamma distribution with parameters α and β has the mean α/β and variance α/β^2. Those variables are used only when usedata = 1 and have no effect when usedata = 2 or 3.

alpha_gamma = 1 1 specifies the shape and scale parameters (α and β) in the gamma prior for the shape parameter for gamma rates among sites in models such as JC69+Γ, K80+Γ etc. The gamma model of rate variation is assumed only if the variable **alpha** is assigned a positive value. This prior is used only when usedata = 1 and has no effect when usedata = 2 or 3.

rgene_gamma = 2 2 specifies the shape and scale parameters (α and β) in the gamma prior for the overall rate parameter μ. Under the global-clock model (clock=1), the independent-rates model (clock = 2), and also the correlated-rates model (clock = 3), μ is the overall rate on the tree specified by the prior. In the example, μ has the prior mean 2/2 = 1, that is, one change per site per time unit. If one time unit is 100MY, this means an overall average rate of 10^{-8} substitutions per site per year. The variance ($2/2^2 = 0.5$ in the example) of this gamma prior specifies how confident you are about the overall rate, or how variable the overall rates are among loci. It is not about how variable the rates are among branches or how wrong the clock is.

You need to adjust this prior to suit your data and the chosen time scale. Don't use the default. A pragmatic way of deriving a rough rate estimate (for use as the prior mean) may be to use baseml or codeml under the global clock model (clock = 1), with point calibrations (as in Yoder and Yang 2000). If the same species are included in every locus, it is quite easy to do this. Otherwise it is a bit more complex, and here is a possible procedure. First run mcmctree with usedata = 3, so that the program generates tmp1.txt, tmp1.ctl, and tmp1.trees. The tree is unrooted. Edit the tree to make it rooted by adding a pair of parentheses. Edit the tree to add one or two point calibrations. If a node has fossil bounds '>0.6<0.8', you can use '=0.7'. Edit tmp1.ctl to add a line clock = 1, and change the value of getSE from 2 to 0. Then run baseml tmp1.ctl and look at the result file out to get the rate. Do the same thing for another locus. This way you will get an idea about the overall rates among loci and how variable they are among loci, information useful for specifying the prior.

sigma2_gamma = 1 1 specifies the shape and scale parameters (α and β) in the gamma prior for parameter σ^2, which specifies how variable the rates are across branches. This prior is used for the two variable-rates models (clock = 2 or 3), with a larger σ^2 indicating more variable rates (Rannala and Yang 2007). If clock = 1, this prior has no effect.

In the independent-rates model (clock = 2), rates for branches are independent variables from a log-normal distribution (Rannala and Yang 2007: equation 9).

$$f(r \mid \mu, \sigma^2) = \frac{1}{r\sqrt{2\pi\sigma^2}} \exp\left\{-\frac{1}{2\sigma^2}\left[\log(r/\mu) + \tfrac{1}{2}\sigma^2\right]^2\right\}, \quad 0 < r < \infty. \qquad (1)$$

Here σ^2 is the variance in the logarithm of the rates. The rate r has mean μ and variance $(e^{\sigma^2} - 1)\mu^2$.

The correlated-rates model (clock = 3) specifies the density of the current rate r, given that the ancestral rate time t ago is r_A, as

$$f(r \mid r_A, t\sigma^2) = \frac{1}{r\sqrt{2\pi t\sigma^2}} \exp\{-\frac{1}{2t\sigma^2}(\log(r/r_A) + \tfrac{1}{2}t\sigma^2)^2\}, \quad 0 < r < \infty \qquad (2)$$

(Rannala and Yang 2007: equation 2). Parameter σ^2 here is equivalent to v in Kishino *et al.* (2001). Thus r has mean r_A and variance $(e^{t\sigma^2} - 1)r_A^2$.

Note that σ^2 (clock = 2) or $t\sigma^2$ (clock = 3) is not the variance of the rate; it is the variance of the logarithm of the rate.

The log normal distribution is not scale-invariant. When one changes the time unit from 10MY to 100MY (so that the rate, per time unit, is 10 times as large), and changes the priors on times and rates accordingly, one may hope that the posterior estimates stay the same. This is not the case. The log normal may look reasonable on one time scale and not on the other. It is advisable to plot the log-normal density using the prior mean for σ^2 and make sure it is not unreasonable.

finetune. The following line in the control file

```
finetune = 0: 0.04 0.2 0.3 0.1 0.3 * time, rate, mixing, paras, RateParas
```

is about the step lengths used in the proposals in the MCMC algorithm. The first value, before the colon, is a switch, with 0 meaning no automatic adjustments by the program and 1 meaning automatic adjustments by the program. Following the colon are the step lengths for the proposals used in the program. The proposals are as follows: (a) to change the divergence times, (b) to change the rates, (c) to perform the mixing step (page 225 in Yang and Rannala 2006), (d) to change parameters in the substitution model (such as κ and α in HKY+Γ), and (e) to change parameters in the rate-drift model (for clock = 2 or 3 only). If you choose to let the program adjust the step lengths, burnin has to be >200, and then the step lengths specified here will be the initial step lengths, and the program will try to adjust them using the information collected during the burnin step. It does this twice, once at half of the burnin and another time at the end of the burnin. The option of automatic adjustment is not well tested.

The following notes are for manually adjusting the steplengths. You can use them to generate good initial step lengths as well for the option of automatic step length adjustment.

```
-20% 0.33 0.01 0.25 0.00 0.00  1.022 0.752 0.252 0.458 0.133 0.843 - 0.074 0.787 -95294.7
-15% 0.33 0.01 0.25 0.00 0.00  1.021 0.751 0.253 0.457 0.130 0.841 - 0.067 0.783 -95295.4
-10% 0.33 0.00 0.26 0.00 0.00  1.022 0.752 0.254 0.458 0.129 0.842 - 0.065 0.781 -95294.6
 -5% 0.33 0.00 0.25 0.00 0.00  1.022 0.751 0.254 0.457 0.128 0.841 - 0.063 0.780 -95292.4
  0% 0.32 0.00 0.25 0.00 0.00  1.022 0.751 0.254 0.457 0.128 0.841 - 0.063 0.780 -95290.2
  2% 0.32 0.00 0.27 0.00 0.00  1.014 0.746 0.253 0.453 0.126 0.833 - 0.059 0.784 -95290.4
```

A few seconds or minutes (hopefully not hours) after you start the program, the screen output will look like the above. The output here is generated from a run under the JC model and global clock (clock = 1). The percentage % indicates the progress of the run, with negative values for the burn-in. Then the five proportions (e.g., 0.33 0.01 0.25 0.00 0.00 on the first line) are the acceptance proportions (P_{jump}) for the corresponding proposals. The optimal acceptance proportions are around 0.3, and you should try to make them fall in the interval (0.2, 0.4) or at least (0.15, 0.7). If the

acceptance proportion is too small (say, <0.10), you decrease the corresponding finetune parameter. If the acceptance proportion is too large (say, >0.80), you increase the corresponding finetune parameter. In the example here, the second acceptance proportion, at 0.01 or 0.00, is too small, so you should stop the program (Ctrl-C) and modify the control file to decrease the corresponding finetune parameter (change 0.2 into 0.02, for example). Then run the program again (use the up ↓ and down ↑ arrow keys to retrieve past commands), observe it for a few seconds or minutes and then kill it again if the proportions are still not good. Repeat this process a few times until every acceptance proportion is reasonable. This is not quite so tedious as it may sound.

The finetune parameters in the control file are in a fixed order and always read by the program even if the concerned proposal is not used (in which case the corresponding finetune parameter has no effect). In the above example, JC does not involve any substitution parameters, so that the 4th finetune parameter has no effect, and the corresponding acceptance proportion is always 0. This proportion is always 0 also when the approximate likelihood calculation is used (usedata = 2) because in that case the likelihood is calculated by fitting the branch lengths to a normal density, ignoring all substitution parameters like κ, α etc. If clock = 1, there are no parameters in the rate-drift model, so that the 5th acceptance proportion is always 0.

Note that the impact of the finetune parameters is on the efficiency of the algorithm, or on how precise the results are when the chain is run for a fixed length. Even if the acceptance proportions are too high or too low, reliable results will be obtained in theory if the chain is run sufficiently long. This effect is different from the effect of the prior, which affects the posterior estimates.

print = 1 means that samples will be taken in the MCMC and written to disk and the posterior results will be summarized. 0 means that the posterior means will be printed on the monitor but nothing else: this is mainly useful for testing the program.

burnin = 2000, sampfreq = 5, nsample = 10000. In the example here, the program will discard the first 2000 iterations as burn-in, and then run the MCMC for 5 × 10000 iterations, sampling (writing to disk) every 5 iterations. The 10000 samples will then be read in and summarized. I think you should take at least 2000 samples.

Differences between MCMCtree and Multidivtime

Here are a few differences between the two programs.

- Soft bounds are used in mcmctree while hard bounds are in multidivtime.

- MCMCTREE does not use outgroups, and the master species tree supplied by the user should be the rooted tree for the ingroup species only. With multidivtime, one runs estbranches on an unrooted tree for both ingroup and outgroup species and then run multidivtime on the rooted tree for ingroup species only.

- Calibration information is supplied by identifying the node numbers on the rooted tree for ingroup species for multidivtime, while it is supplied as node labels in the rooted tree in mcmctree.

- In multidivtime, the likelihood is calculated using a normal approximation to the branch lengths in the rooted ingroup tree (see Yang 2006, figure 7.10a). In mcmctree, the likelihood is calculated either exactly or approximately for nucleotide sequences, and approximately for

amino acid or codon sequences. The approximation in mcmctree is applied to the branch lengths in the unrooted tree of the ingroup species (see Yang 2006, figure 7.10b).

- Both programs use a few gamma priors, but the specification is different, with multidivtime requiring the mean (m) and standard deviation (s) and mcmctree using the shape (α) and scale (β) parameters. There are related as follows.

$$m = \alpha/\beta, \quad s = \sqrt{\alpha}/\beta,$$
$$\alpha = (m/s)^2, \quad \beta = m/s^2.$$

- The rate-drift model used in multidivtime is the same as the correlated-rates model (clock = 3) in MCMCtree. The following variables are equivalent in the two programs.

Prior	Multidivtime (`multicntrl.dat`)	MCMCtree (`MCMCtree.ctl`)
Gamma prior on root rate	`rtrate` & `rtratesd`	`mu_gamma`
Gamma prior on ν or σ^2	`brownmean` & `brownsd`	`sigma2_gamma`
Gamma prior on root age	`rttm` & `rttmsd`	Absent

In both programs, a gamma prior is assigned on the root rate (overall rate on the tree), and on parameter ν or σ^2. Multidivtime also assigns a gamma prior on the age of the root, which is not used in MCMCtree.

Approximate likelihood calculation

Thorne *et al.* (1998) suggested the use of the multivariate normal distribution of MLEs of branch lengths to approximate the likelihood function. To implement this approximation, one has to obtain the MLEs of the branch lengths and calculate their variance-covariance matrix or equivalently the matrix of second derivatives of the log likelihood with respect to the branch lengths (this matrix is also called the Hessian matrix). In Thorne's multidivtime package, this is achieved using the program estbranches.

I have implemented this approximation using the option usedata = 3. With this option, mcmctree will prepare three temporary files for each locus and then invoke baseml or codeml to calculate the MLEs of branch lengths and the Hessian matrix. These results are generated in the file rst1 and copied into the file out.BV by mcmctree. The three temporary files for each locus are the control file tmp#.ctl, the sequence alignment tmp#.txt, and the tree file tmp#.trees, where # means the index for the locus. The tree for the locus is generated by mcmctree by pruning the master tree of all species so that only those species present at the locus remain, and by de-rooting the resulting tree. You should not edit this tree file. You can edit the control file tmp#.ctl to use another model implemented in baseml or codeml, and this option should allow you to use amino acid or codon substitution models. The calculation of the Hessian matrix may be sensitive to the step length used in the difference approximation, and it is advisable that you change the variable Small_Diff in the control file tmp#.ctl to see whether the results are stable.

The output file out.BV from usedata = 3 should then be renamed in.BV. This file has one block of results for each locus. If you manually edit the control file tmp#.ctl and then invoke baseml or codeml from the command line (for example, by typing codeml tmp2.ctl), you will have to manually copy the content of rst1 into in.BV.

With usedata = 2, mcmctree will read the MLEs and Hessian matrix from in.BV and apply the approximate method for calculating the likelihood in the MCMC.

In effect, mcmctree/usedata = 3 performs the function of estbranches and you can manually perform this step by running baseml or codeml externally after the tree file tmp#.ctl is generated. Similarly mcmctree/usedata = 2 performs the function of mlutidivtime.

Models of amino acid or codon substitution are not implemented in the mcmctree program for the exact likelihood calculation. The only way to use those models is through the approximate method (usedata = 3 and 2). If is advisable that you edit the intermediate control file tmp#.ctl to choose the appropriate model of amino acid or codon substitution in the codeml analysis, and then copy the results into the in.BV file. Also have a look at the estimated branch lengths in the tree. If many of them are near 0, you should be concerned as perhaps you have too little data or the tree is wrong for the locus. Finally run mcmctree/usedata = 2.

In the description here, a gene or locus means a site partition. For example, since the three codon positions typically have very different rates, different base compositions, etc., you may treat them as separate partitions.